Estrutura de repetição

Uma estrutura de repetição, em programação, é uma construção que permite que um conjunto de instruções seja executado repetidamente enquanto uma condição específica for atendida.

Essa condição é chamada de "**condição de parada**" e é verificada antes de cada execução do conjunto de instruções. Quando a condição não é mais verdadeira, a repetição para.

As estruturas de repetição são fundamentais para automatizar tarefas que precisam ser executadas várias vezes sem a necessidade de escrever o mesmo código repetidamente. Elas permitem que você crie programas mais eficientes e concisos.

Estrutura de repetição FOR

Na estrutura de repetição FOR um conjunto de instruções é executado enquanto uma condição for satisfeita

Sintaxe:

```
for (inicializacao; condicao; incremento) {
  // conjunto de instruções
}
```

Estrutura de repetição WHILE

Na estrutura de repetição **WHILE** a condição é verificada antes da execução do conjunto de instruções. Se a condição já for falsa desde o início, o conjunto de instruções pode nunca ser executado.

Sintaxe:

```
while (condicao) {
  // conjunto de instruções
}
```

Estrutura de repetição DO-WHILE

A estrutura de repetição **DO-WHILE** é semelhante ao **WHILE**, mas com uma diferença importante: a condição é verificada após a execução do conjunto de instruções.

Isso significa que o conjunto de instruções sempre será executado pelo menos uma vez, mesmo que a condição seja inicialmente falsa.

Sintaxe:

```
do {
```

```
 // Conjunto de instruções
} while (condicao);
```

O conjunto de instruções é executado primeiro, independentemente da condição. Após a execução do conjunto de instruções, a condição é verificada. Se a condição for verdadeira, o conjunto de instruções será executado novamente. Isso continuará até que a condição se torne falsa. Se a condição for falsa desde o início, o conjunto de instruções ainda será executado uma vez.

A estrutura DO-WHILE é útil quando você deseja garantir que um conjunto de instruções seja executado pelo menos uma vez, independentemente da condição inicial.

As estruturas de repetição são essenciais para realizar tarefas como processar elementos em uma lista (array), ler dados de um arquivo, criar jogos, realizar cálculos complexos e muito mais. Elas são uma parte fundamental da programação e permitem que os programas realizem ações repetitivas de maneira eficiente.

Estrutura de repetição FOR com mais detalhes

A estrutura de repetição ***FOR*** é uma das estruturas mais comuns em programação e é usada para executar um conjunto de instruções repetidamente por um número específico de vezes. A estrutura ***FOR*** é especialmente útil quando você sabe antecipadamente quantas vezes deseja repetir um bloco de código.

Aqui está a sintaxe básica da estrutura "for" em JavaScript:

```
for (inicializacao; condicao; incremento ou decremento) {
 // conjunto de instruções
}
```

Onde:

- inicialização: É onde você define uma variável e atribui um valor inicial. Essa parte é executada apenas uma vez, no início do loop.
- condição: É uma expressão que é avaliada antes de cada iteração do loop. Se a condição for verdadeira, o conjunto de instruções dentro do loop será executado. Se for falsa, o loop será interrompido.
- Incremento ou decremento: É onde você atualiza a variável definida na inicialização. Geralmente, isso envolve incrementar ou decrementar o valor da variável. Esta parte é executada após cada iteração da estrutura.
- Conjunto de instruções: É o bloco de código que será executado repetidamente enquanto a condição for verdadeira.

Exemplo:

```
for (let i = 1; i <= 5; i++) {
  console.log(i);
}
```

Neste exemplo:

- A variável **i** é inicializada com o valor 1.
- A condição verifica se **i** é menor ou igual a 5. Se for verdadeira, o loop continua.
- Após cada iteração, **i** é incrementado em 1.
- O conjunto de instruções dentro do loop exibe o valor de **i**.

O resultado deste loop será:

```
1
2
3
4
5
```

A estrutura ***FOR*** é uma estrutura de repetição muito versátil e pode ser usada em várias situações, como percorrer elementos de uma lista (array), gerar sequências de números, processar dados em lotes e muito mais. Eles fornecem um controle preciso sobre o número de iterações, tornando-os uma ferramenta fundamental na programação.

Estrutura de repetição WHILE

A estrutura de repetição **while** é utilizada para executar um bloco de código repetidamente enquanto uma condição especificada for verdadeira.

A sintaxe do **while** é a seguinte:

```
while (condicao) {
  // código a ser executado enquanto a condição for verdadeira
}
```

O bloco de código dentro das chaves “{}” é repetidamente executado enquanto a condição fornecida dentro dos parênteses “()” for verdadeira.

A condição é avaliada antes da execução do bloco de código. Se a condição for falsa desde o início, o bloco de código nunca será executado.

Aqui temos um exemplo simples da estrutura **while** que imprime os números de 1 a 5:

```
let contador = 1;

while (contador <= 5) {
  console.log(contador);
```

```
  contador++;
}
```

Neste exemplo a variável **contador** é inicializada com 1.

A condição **contador <= 5** é avaliada antes de cada execução do bloco de código e enquanto a condição for verdadeira, o bloco de código é executado.

Dentro do bloco de código, o valor da variável **contador** é impresso e então a variável tem um incremento (**contador++**) para que a próxima iteração tenha um valor diferente.

Este loop continuará a ser executado até que a condição **contador <= 5** seja falsa. Uma vez que **contador** atinge o valor 6, a condição torna-se falsa, e o loop é encerrado.

É importante garantir que a condição eventualmente se torne falsa para evitar loops infinitos.

Exemplo de um loop infinito (evite fazer isso!)

```
while (true) {
  console.log("Este é um loop infinito!");
}
```

No exemplo acima, a condição é sempre *true*, então o loop continuaria indefinidamente, o que não é desejável na maioria dos casos. Portanto, certifique-se de definir condições que eventualmente se tornem falsas para garantir que o loop tenha uma conclusão.

Estrutura de repetição DO-WHILE

A estrutura de repetição **do-while** é uma forma de loop que executa um bloco de código enquanto uma condição especificada for verdadeira.

A diferença principal entre **do-while** e **while** é que o bloco de código dentro de **do-while** é executado pelo menos uma vez, mesmo que a condição seja falsa desde o início.

A sintaxe é a seguinte:

```
do {
  // código a ser executado
} while (condicao);
```

Exemplo:

```
let contador = 1;

do {
  console.log("Contagem: " + contador);
```

```
  contador++;
} while (contador <= 5);
```

Neste exemplo, o código dentro do bloco "**do**" será executado pelo menos uma vez, mesmo que a condição **contador <= 5** seja falsa desde o início.

Agora, vamos comparar **do-while** com a estrutura **while**:

Execução do Bloco:

Em **while**, o bloco de código só é executado se a condição for verdadeira desde o início.

Em **do-while**, o bloco de código é executado pelo menos uma vez, independentemente da condição ser verdadeira ou falsa inicialmente.

Avaliação da Condição:

Em **while**, a condição é avaliada antes da execução do bloco, o que significa que o bloco pode não ser executado se a condição for falsa desde o início.

Em **do-while**, a condição é avaliada após a execução do bloco, garantindo que o bloco seja executado pelo menos uma vez.

Quando usar **do-while** e quando usar **while**:

Use o **do-while** quando você deseja garantir que o bloco de código seja executado pelo menos uma vez, independentemente da condição inicial, neste caso a lógica dentro do bloco será executada antes de verificar a condição.

Use **while** quando você deseja executar o bloco de código apenas se a condição for verdadeira desde o início, neste caso a lógica dentro do bloco depende da condição ser verdadeira para ser executada.

Um cenário comum em que a estrutura de repetição **do-while** pode ser mais apropriada é quando você precisa solicitar a entrada de dados pelo menos uma vez e, em seguida, continuar a solicitar essa entrada até que ela atenda uma condição específica.

Vamos ver um exemplo em que estamos solicitando ao usuário para fornecer um número maior que 5:

```
let userInput;

do {

  userInput = prompt("Digite um número maior que 5: ");
  userInput = parseInt(userInput);  // Converte a entrada para um número inteiro

} while (isNaN(userInput) || userInput <= 5);
```

```
console.log("Número válido: " + userInput);
```

Neste exemplo:

O bloco dentro de “**do**” solicita ao usuário para digitar um número.

A entrada do usuário é convertida para um número inteiro usando **parseInt**.

O loop **do-while** continuará a pedir que o usuário digite um novo número enquanto a entrada não for um número válido (verificado por **isNaN**) ou enquanto o número é menor ou igual a 5.

Este é um cenário onde a estrutura **do-while** é útil, pois garante que o bloco de código seja executado pelo menos uma vez, independentemente da validade da entrada inicial do usuário.

Se estivéssemos usando **while** puro, e a entrada inicial do usuário não fosse válida, o bloco de código não seria executado, o que pode não ser desejado em certas situações.

Break

O **Break** abandona o laço.

Observe:

```
let numeros = ''

for (let i = 1; i <= 100; i++) {
   numeros += i + ' - '

   if (i % 11 == 0) {
      break;
   }
}
console.log(numeros)
```

Continue

O **continue** interrompe somente o laço atual permitindo o fluxo de execução da iteração continue.

Observe:

```
let numeros = ''

for (let i = 1; i <= 100; i++) {
   if (i % 2 == 0) {
      continue
   }
   numeros += i + ' - '
```

```
}

console.log(numeros)
```

Exercícios:

Exercício 63:

Crie uma aplicação que receba três números.

Exibir os três números informando se eles são positivos, negativos ou nulos (considere o número nulo se ele for igual a zero).

Informar o maior número.

Exercício 64:

Crie uma aplicação que receba dois valores numéricos e efetue sua adição.

Caso o resultado da adição seja maior que 10, exibir os números digitados, o valor da adição e a raiz cúbica da adição.

Caso contrário exibir somente os valores digitados e o valor da adição.

Exercício 65:

Crie uma aplicação que faça a leitura de três valores e apresente como resultado final a soma dos quadrados dos três valores lidos. Apresentar também se a soma é um número par ou ímpar.

Exercícios de análise:

Você deve ler e interpretar os exercícios, depois selecione a alternativa correta.

Observação:

- Não execute o código
- Olhe a resposta somente após a execução do exercício

Exercício 66:

Analise o código e assinale a alternativa correta:

```
let numero = 5;

if (numero % 2 = 0) {
  console.log("Par");
} else {
```

```
  console.log("Ímpar");
}
```

Aqui estão as alternativas:

A) O código está correto. Ele vai imprimir "Ímpar" quando numero for ímpar e "Par" quando numero for par.

B) O operador = deve ser substituído por === na linha if (numero % 2 = 0).

C) A variável numero deve ser declarada com var em vez de let.

D) A variável numero deve ser inicializada com um número decimal, como let numero = 2.5, para testar corretamente se o código funciona.

E) A variável numero deve ser inicializada com uma string, como let numero = "5", para testar corretamente se o código funciona.

Resposta exercício 66:

A resposta correta é a alternativa B.

Explicação:

O código contém um erro de sintaxe na linha if (numero % 2 = 0). O operador de comparação de igualdade deve ser === em vez de =.

Portanto, a linha correta deve ser if (numero % 2 === 0), para verificar se numero é divisível por 2 e, assim, determinar se é par ou ímpar. O código funcionará corretamente após a correção desse erro.

Exercício 67:

Analise o código e assinale a alternativa correta:

```
let numero = -7;

if (numero > 0) {
  console.log("Positivo");
} else {
  console.log("Negativo");
}
```

Aqui estão as alternativas:

A) O código está correto. Ele vai imprimir "Negativo" quando numero for negativo e "Positivo" quando numero for positivo.

B) O operador > deve ser substituído por === na linha if (numero > 0).

C) A variável numero deve ser declarada com var em vez de let.

D) A variável numero deve ser inicializada com um número decimal, como let numero = -3.5, para testar corretamente se o código funciona.

E) A variável numero deve ser inicializada com uma string, como let numero = "-7", para testar corretamente se o código funciona.

Resposta exercício 67:

A resposta correta é a alternativa A.

Explicação:

O código está correto e funcionará como esperado. Ele verifica se número é maior que 0 e, se for, imprime "Positivo"; caso contrário, imprime "Negativo".

Exercício 68:

Analise o código e assinale a alternativa correta:

Você está criando um programa para calcular se uma pessoa tem um índice de massa corporal (IMC) saudável com base em seu peso e altura. Abaixo está o código do programa que você escreveu diretamente no programa principal. Qual é a alternativa correta para determinar se o código funciona corretamente?

```
let peso = 70;
let altura = 1.75;
let imc = peso / (altura * altura);

if (imc >= 18.5 && imc <= 24.9) {
  console.log("IMC saudável");
} else {
  console.log("IMC não saudável");
}
```

Aqui estão as alternativas:

A) O código está correto. Ele vai imprimir "IMC saudável" quando o IMC estiver entre 18.5 e 24.9 (inclusive), caso contrário, imprimirá "IMC não saudável".

B) O operador <= deve ser substituído por < na linha if (imc >= 18.5 && imc <= 24.9).

C) A variável altura deve ser inicializada com um número inteiro, como let altura = 175, para testar corretamente se o código funciona.

D) Todas as alternativas estão erradas.

E) A variável peso deve ser inicializada com uma string, como let peso = "70", para testar corretamente se o código funciona.

Resposta exercício 68:

A resposta correta é a alternativa A.

Explicação:

O código está correto e funcionará como esperado. Ele calcula o IMC com base no peso e na altura, e em seguida, verifica se o IMC está dentro da faixa

considerada saudável (entre 18.5 e 24.9, inclusos) e imprime "IMC saudável" ou "IMC não saudável" com base nessa verificação.

Exercício 69:

Você está criando um programa para contar quantas vezes uma pessoa clica em um botão em um site. O programa deve registrar o número atual de cliques e, em seguida, incrementar esse número sempre que o botão for clicado. Abaixo está o código que você escreveu:

```
let numCliques = 0;

// Simulação de clique no botão
numCliques++;

// Simulação de mais cliques no botão
numCliques += 2;

console.log("Número de cliques: " + numCliques);
```

Qual será o valor impresso no console após a execução deste código?

A) O valor impresso será "Número de cliques: 0".

B) O valor impresso será "Número de cliques: 1".

C) O valor impresso será "Número de cliques: 2".

D) O valor impresso será "Número de cliques: 3".

E) O valor impresso será "Número de cliques: 4".

Resposta exercício 69:

A resposta correta é a alternativa D.

Explicação:

O código começa com numCliques inicializado como 0. Em seguida, ele incrementa numCliques usando o operador ++ (que incrementa o valor em 1) e depois adiciona 2 ao valor de numCliques usando += 2. Portanto, após essas operações, numCliques será igual a 3. O valor impresso no console será "Número de cliques: 3".

Exercício 70:

Você está criando um programa para registrar a quantidade de itens vendidos em uma loja. O programa deve inicializar o contador de itens vendidos como zero e, em seguida, incrementá-lo sempre que um item for vendido. Abaixo está o código que você escreveu:

```
let itensVendidos = 0;
```

```
// Simulação de venda
itensVendidos += 3;

// Simulação de venda
itensVendidos += 2;

console.log("Itens vendidos: " + itensVendidos);
```

Qual será o valor impresso no console após a execução deste código?

A) O valor impresso será "Itens vendidos: 0".

B) O valor impresso será "Itens vendidos: 1".

C) O valor impresso será "Itens vendidos: 3".

D) O valor impresso será "Itens vendidos: 5".

E) O valor impresso será "Itens vendidos: 6".

Resposta exercício 70:

A resposta correta é a alternativa D.

Explicação:

O código começa com itensVendidos inicializado como 0. Em seguida, ele incrementa itensVendidos adicionando 3 e depois mais 2. Portanto, após essas operações, itensVendidos será igual a 5. O valor impresso no console será "Itens vendidos: 5".

Exercício 71:

Você está desenvolvendo um programa que controla o estoque de produtos em uma loja. O programa inicializa a quantidade de um produto em estoque e, em seguida, decrementa essa quantidade sempre que um produto é vendido. Abaixo está o código que você escreveu:

```
let estoqueProdutoA = 10;
estoqueProdutoA -= 3;
estoqueProdutoA -= 2;

console.log("Quantidade de Produto A em estoque: " + estoqueProdutoA);
```

Qual será o valor impresso no console após a execução deste código?

A) O valor impresso será "Quantidade de Produto A em estoque: 10".

B) O valor impresso será "Quantidade de Produto A em estoque: 7".

C) O valor impresso será "Quantidade de Produto A em estoque: 5".

D) O valor impresso será "Quantidade de Produto A em estoque: 3".

E) O valor impresso será "Quantidade de Produto A em estoque: 0".

Resposta exercício 71:

A resposta correta é a alternativa C.

Explicação:

O código começa com estoqueProdutoA inicializado como 10. Em seguida, ele decrementa estoqueProdutoA subtraindo 3 unidades e depois mais 2 unidades. Portanto, após essas operações, estoqueProdutoA será igual a 5. O valor impresso no console será "Quantidade de Produto A em estoque: 5".

Exercício 72:

Você está desenvolvendo um programa que monitora a contagem de pessoas em uma sala de conferências. O programa começa com uma contagem inicial e, em seguida, intercala a entrada e a saída de pessoas usando os operadores de incremento ++, += e decremento --, -=. Abaixo está o código que você escreveu:

```
let contagemPessoas = 0;

contagemPessoas += 3;
contagemPessoas--;
contagemPessoas += 2;
contagemPessoas--;

console.log("Número de pessoas na sala: " + contagemPessoas);
```

Qual será o valor impresso no console após a execução deste código?

A) O valor impresso será "Número de pessoas na sala: 0".

B) O valor impresso será "Número de pessoas na sala: 2".

C) O valor impresso será "Número de pessoas na sala: 3".

D) O valor impresso será "Número de pessoas na sala: 4".

E) O valor impresso será "Número de pessoas na sala: 5".

F) Todas as alternativas estão erradas.

Resposta exercício 72:

A resposta correta é a alternativa C.

Explicação:

O código começa com contagemPessoas inicializado como 0. Em seguida, ele incrementa contagemPessoas adicionando 3 pessoas, depois decrementa 1 pessoa, adiciona mais 2 pessoas e, por fim, decrementa 1 pessoa. Portanto, após essas operações, contagemPessoas será igual a 3. O valor impresso no console será "Número de pessoas na sala: 3".

Exercícios

Exercício 73:

Crie uma aplicação para imprimir a soma de todos os números de 0 à 100.

Exercício 74:

Escreva um programa que exiba todos os números pares de 1 a 50.

Exercício 75:

Escreva um programa que calcule o fatorial de um número fornecido pelo usuário.

O fatorial de um número é um conceito matemático usado para calcular o produto de todos os números inteiros positivos de 1 até esse número. É denotado pelo símbolo "!".

O fatorial de um número "n" é calculado da seguinte forma:

$$n! = n \times (n - 1) \times (n - 2) \times ... \times 2 \times 1$$

Por exemplo:

- O fatorial de 5 é calculado como $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$.
- O fatorial de 4 é calculado como $4! = 4 \times 3 \times 2 \times 1 = 24$.
- O fatorial de 0 (zero) é por definição 1.

Exercício 76:

Escreva um programa que gere uma sequência de números a partir de um número inicial fornecido pelo usuário e exiba os primeiros 10 números dessa sequência.

Exemplo:

Número fornecido: 5
Sequência: 5, 10, 15, 20, 25, 30, 35, 40, 45, 50

Exercício 77:

Escreva um programa que exiba a tabuada de um número fornecido pelo usuário.

Exercício 78:

Escreva um programa que exiba os números de 1 a 20. Para cada número, verifique se ele é par ou ímpar e exiba essa informação.

Exercício 79:

Escreva um programa que solicite ao usuário que insira 5 notas e, em seguida, calcule e exiba a média das notas.

Exercício 80:

Escreva um programa que solicite ao usuário que insira 10 números.

Este programa deve contar quantos números são positivos e maiores que zero, contar quantos números são negativos e exiba os resultados.

Exercício 81:

Crie uma aplicação que receba cinco números e imprima o quadrado de cada número.